Security Models
Trusted Zones
Access Control

Slides credit to Ethan L. Miller and Scott A. Brandt
Protection Domains

- Three protection domains
  - Each lists objects with permitted operations
- Domains can share objects & permissions
  - Objects can have different permissions in different domains
  - There need be no overlap between object permissions in different domains
- How can this arrangement be specified more formally?

- File1 [R]
- File2 [RW]
- File3 [R]
- File4 [RWX]
- File5 [RW]
- File3 [W]
- Printer [W]
- Screen1 [W]
- Mouse [R]
### Access Matrix

<table>
<thead>
<tr>
<th>Domain</th>
<th>File1</th>
<th>File2</th>
<th>File3</th>
<th>File4</th>
<th>File5</th>
<th>Printer1</th>
<th>Mouse</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Read</td>
<td>Read</td>
<td>Write</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>Read</td>
<td></td>
<td>Read</td>
<td>Write</td>
<td></td>
<td>Write</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>Write</td>
<td></td>
<td></td>
<td></td>
<td>Write</td>
<td>Read</td>
</tr>
</tbody>
</table>

- Each domain has a row in the matrix
- Each object has a column in the matrix
- Entry for `<object,column>` has the permissions
  - Who’s allowed to modify the protection matrix? What changes can they make?
- How is this implemented efficiently?
Representing the protection matrix

- Need to find an efficient representation of the protection matrix (also called the access matrix)
- Most entries in the matrix are empty!
- Compress the matrix by:
  - Associating permissions with each object → access control list
  - Associating permissions with each domain → capabilities
- How is this done, and what are the tradeoffs?
Access control lists

- Each object has a list
  - Protection domain
    - User name
    - Group of users
  - Access rights
    - Read
    - Write
    - Execute (?)
- No entry for domain => no rights for that domain
- Operating system checks permissions when access is needed

File1
- elm: <R,W>
- znm: <R>
- root: <R,W,X>

File2
- elm: <R,X>
- uber: <R,W>
- root: <R,W>
- all: <R>
Access control lists in the real world

- Unix file system
  - Access list for each file has exactly three domains on it
    - User (owner), Group, Others
  - Rights: read, write, execute

- AFS (Andrew File System) for client/server systems
  - Access lists only apply to directories
  - Files inherit rights from the directory they are in
  - Access list may have many entries on it with possible rights:
    - read, write, lock (for files in the directory)
    - lookup, insert, delete (for the directories themselves),
    - administer (ability to add or remove rights from the ACL)
Capabilities

- Each process has a capability list
- List: one entry per object the process can access
  - Object name
  - Object permissions

- Objects not listed are not accessible

- How are these secured?
  - Kept in kernel
  - Cryptographically secured

```
<table>
<thead>
<tr>
<th></th>
<th>File 1</th>
<th>File 2</th>
<th>File 3</th>
<th>File 4</th>
<th>File 7</th>
<th>File 9</th>
</tr>
</thead>
</table>
```
Protecting the access matrix: summary

- OS must ensure that the access matrix isn’t modified (or even accessed) in an unauthorized way.

- Access control lists
  - Reading or modifying the ACL is a system call
  - OS makes sure the desired operation is allowed

- Capability lists
  - Similar to ACLs: reading and modification done by OS
  - Can be handed to processes and verified cryptographically later on
  - May be better for widely distributed systems where capabilities can’t be centrally checked
Security models
for the confidentiality and integrity of files
**Multi-Level Security (MLS)**

Military security policy is based on protecting classified information

Hierarchy of Sensitivity ranks

- Unclassified
- Restricted
- Confidential
- Secret
- Top Secret

Least Sensitive

Most Sensitive
Compartments and Sensitivity ranks

Compartments may not be hierarchical
**Need-to-know Rule In Military Security Policy**

- \(<\text{rank, compartments}>\) class or classification of information
- A **clearance** is an indication that a person is trusted to access data up to some rank
- **Dominance relation** \(\geq\) between subjects and objects
  - \(s\) dominates \(o\) (denoted by \(s \geq o\))

For subject \(s\) and object \(o\),
\[
s \geq o \text{ if and only if } \begin{cases} \text{rank}_s \geq \text{rank}_o \\ \text{compartments}_o \supseteq \text{compartments}_s \end{cases}
\]

\(s\) has a need to know all the compartments for which \(o\) is classified

A subject \(s\) can read an object \(o\) only if \(s\) dominates \(o\)

- e.g., top secret > secret
- Bell-LaPadula – confidentiality model (BLP)
- Biba – integrity model
Bell-LaPadula Model - read down/write up

- Security levels arranged in linear ordering
  - Top Secret: highest
  - Secret
  - Confidential
  - Unclassified: lowest

<table>
<thead>
<tr>
<th>security level</th>
<th>subject</th>
<th>object</th>
</tr>
</thead>
<tbody>
<tr>
<td>Top Secret</td>
<td>Tamara</td>
<td>Personnel Files</td>
</tr>
<tr>
<td>Secret</td>
<td>Samuel</td>
<td>E-Mail Files</td>
</tr>
<tr>
<td>Confidential</td>
<td>Claire</td>
<td>Activity Logs</td>
</tr>
<tr>
<td>Unclassified</td>
<td>Alice</td>
<td>Telephone Lists</td>
</tr>
</tbody>
</table>

- Levels consist of security clearance $L(s)$
  - Objects have security classification $L(o)$

- Tamara can read all files
- Claire cannot read Personnel or E-Mail Files
- Alice can only read Telephone Lists
Reading/Writing Information

- Information flows *up*, not *down*
  - “Reads up” disallowed, “reads down” allowed
  - Subject \( s \) can read object \( o \) iff \( L(o) \leq L(s) \) and \( s \) has permission to read \( o \) – (aka simple security condition)
  - “No reads up” rule

“ Writes up” allowed, “writes down” disallowed

*-Property*
Subject \( s \) can write object \( o \) iff \( L(s) \leq L(o) \) and \( s \) has permission to write \( o \). (aka *-property)
“No writes down” rule
Basic Security Theorem in BLP model

- If a system is initially in a secure state, and every transition of the system satisfies the simple security condition, and the *-property, then every state of the system is secure.
What’s Wrong with BLP?
Biba multilevel integrity model

- Principles to guarantee integrity of data
- Simple integrity principle
  - A process can write only objects at its security level or lower
  - No way to plant fake information at a higher level
- The integrity * property
  - A process can read only objects at its security level or higher
- Biba is in direct conflict with Bell-La Padula
  - Difficult to implement both at the same time!
Biba Integrity Model

- Bell-La Padula model (read-down/write-up)

- Biba (read-up/write-down) – dual of BLP model
  - Subject cannot corrupt data in a higher level
  - Subject cannot be corrupted by data with a lower level
  1. \( s \in S \) can read \( o \in O \) iff \( i(s) \leq i(o) \)
  2. \( s \in S \) can write to \( o \in O \) iff \( i(o) \leq i(s) \)
  3. \( s_1 \in S \) can execute \( s_2 \in S \) iff \( i(s_2) \leq i(s_1) \)

- \( i \) is integrity level
Key Points

- MAC vs. DAC (mandatory access control vs. discretionary access control)
- Confidentiality models restrict flow of information
- Bell-LaPadula models multilevel security
  - Cornerstone of much work in computer security
- BLP model is generally believed to be very restrictive
- Integrity policies deal with trust
  - As trust is hard to quantify, hard to evaluate completely
  - Look for assumptions and trusted users to find possible weak points
- Clark-Wilson (another model) focuses on separation of duty and transactions